

CLIENT SIDE PAGE EDITOR

EVERETT KENYON LOCKHART

Bachelor of Science

Tougaloo College

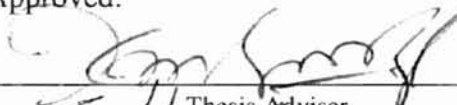
Tougaloo, Mississippi

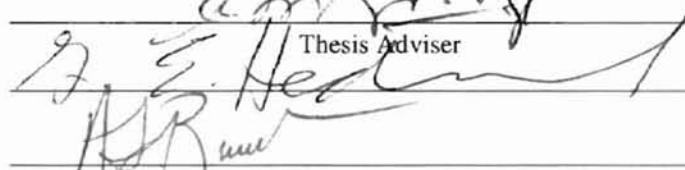
1997


Submitted to the Faculty of the
Graduate College of
Oklahoma State University
In partial fulfillment of
The requirements for
The Degree of
MASTER OF SCIENCE
December, 2000

CLIENT SIDE PAGE EDITOR

Thesis Approved:



Thesis Adviser




Dean of the Graduate College

PREFACE

Developing World Wide Web, WWW, editing applications requires the programmer to know and understand a variety of web-developing languages and editing tools. This thesis introduces WEB editors to familiarize the reader with the topic discussed. However, to fully appreciate the developed product, the reader should obtain a practical experience with WEB editors and the Internet. This thesis is organized in a way that even the novice user can understand and follow. Therefore, it is conceivable for readers to walk away with a general, but complete, grasp of the topic presented. This thesis exposes the reader to basic editors available, their differences, and the features offered by each respective editor. After establishing a sound foundation for the subject, the Client Side Page editor is introduced. The design, implementation, and functionality of the Client Side Page editor are covered to provide the reader with a complete conception of the presented work.

ACKNOWLEDGMENTS

Finally, I would like to thank everyone who has been with me on this long journey. Dr. George, my advisor and friend, without him none of this would have been possible. I would also like to thank my other committee members, Dr. Burrell and Dr. Hedrick for their views on various aspects of my work. My mother and father for their patience in waiting for me to finish school and finally get a job. Tony and Robert for getting me over that one hump which gave me problems that only they could solve. Kerry for traveling right along with me, from start to finish, through those long hours in the office and labs. All of my professors, for it was you who taught me what I know and how to learn more when the situation demands. Leegand, or should I say Dr. Burge III, you taught me more than you will ever know. Ralph, Billy, and all of my friends that have been there throughout, I thank you also. Bradley, I cannot forget you. You have taught me more practical material than anyone has. Most of all, I would like to thank the Lord for allowing me to succeed at yet another level in my life.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.....	1
Thesis Organization.....	3
II. RELATED WORKS.....	4
Current Web Document Structure.....	4
Current Page Break Solutions.....	5
Web Editors.....	9
III. CSE STRUCTURE & IMPLEMENTATION.....	19
Design.....	19
Interface.....	21
Environment.....	24
Capturing & Modifying Document Text.....	24
Functionality.....	25
Security.....	26
IV. CONCLUSION AND FUTURE WORK.....	28
BIBLIOGRAPHY.....	29

LIST OF FIGURES

Figure	Page
1. CSE Structural Diagram.....	2
2. HTML Anchor Technique (Web View).....	6
3. HTML Anchor Technique (Code View).....	7
4. CSS Page Break Technique (Code View).....	8
5. CSS Page Break Technique (Web View)	9
6. FrontPage Editor Interface (Code View).....	11
7. FrontPage Editor Interface (WYSIWYG View).....	11
8. HotDog Editor Interface.....	13
9. Netscape Composer Interface (WYSIWYG View).....	15
10. Netscape Composer Interface (Code View).....	16
11. Dynamic Page Editor Interface (DPE).....	18
12. CSE Infrastructure	20
13. MS Internet Explorer Interface	22
14. CSE Interface	23
15. Sample Code for Positioning Toolbars.....	23
16. Sample Code for Toolbar Creation	23
17. Sample Code for Recognizing Break Location	26

NOMENCLATURE

ASP	active server pages
CSE	client side page editor
CSS	cascading style sheets
CSSP	cascading style sheet positioning
DHTML	dynamic hypertext markup language
DPE	dynamic page editor
HTML	hypertext markup language
IE	Internet Explorer
MSIE	Micro Soft Internet Explorer
URL	universal resource locator
VB Script	Visual Basic Script
WEB	World Wide Web
WWW	World Wide Web
WYSIWYG	what you see is what you get

Chapter I

1.0 Introduction

WEB documents are the central component of World Wide Web (WWW) applications today. Due to the increasing usage of the Internet along with mass production of WEB sites, pages, etc., the need for useful and efficient WEB editing tools has grown. In an attempt to satisfy this need, WEB application developers have developed many useful editing applications. These applications are targeted towards a variety of users, and they have various features that provide them with their own degree of uniqueness [22]. Microsoft's FrontPage editor and Netscape's Composer are examples of such general-purpose editors. Users now have the freedom to choose the type of editor that satisfies their needs. These editors are available in either single or multi-view editing modes. Some have features that are defined specifically for their platform, and others simply provide the basic features needed for WEB document editing. There are several editors available for users to download either commercially or in the public domain. Some of the editing tools featured by these editors include: cut, paste, copy, changing text style and font size, basic html support, etc. However, in their continued effort to advance technology pertaining to WEB page development and maintenance, the page break feature has been *overlooked* and left without extensive development. This feature is very important to users who want only to print and view selected sections of a page. In addition to the page-break feature, the editor described in this thesis can provide other useful features that are discussed in later sections.

In this thesis, we develop an editor that provides the interface necessary to include page breaks and to modify a WEB document at viewing time. The editor defined page breaks will allow the document to be broken into multiple pages for viewing and printing. Initially, Joongseok Park addressed this problem in his thesis [11]. This thesis builds on the work presented by J. Park. Through the use of DHTML, CSS, and JavaScript, this WEB editor has the dynamic capability to break a document into pages for printing and/or viewing purposes. It presents the user with the freedom to choose any location where he/she feels a page break should occur. It produces results almost immediately. Figure 1.1 shows the overall placement of the CSE (Client-Side page editor) in relation to the browser and WEB components.

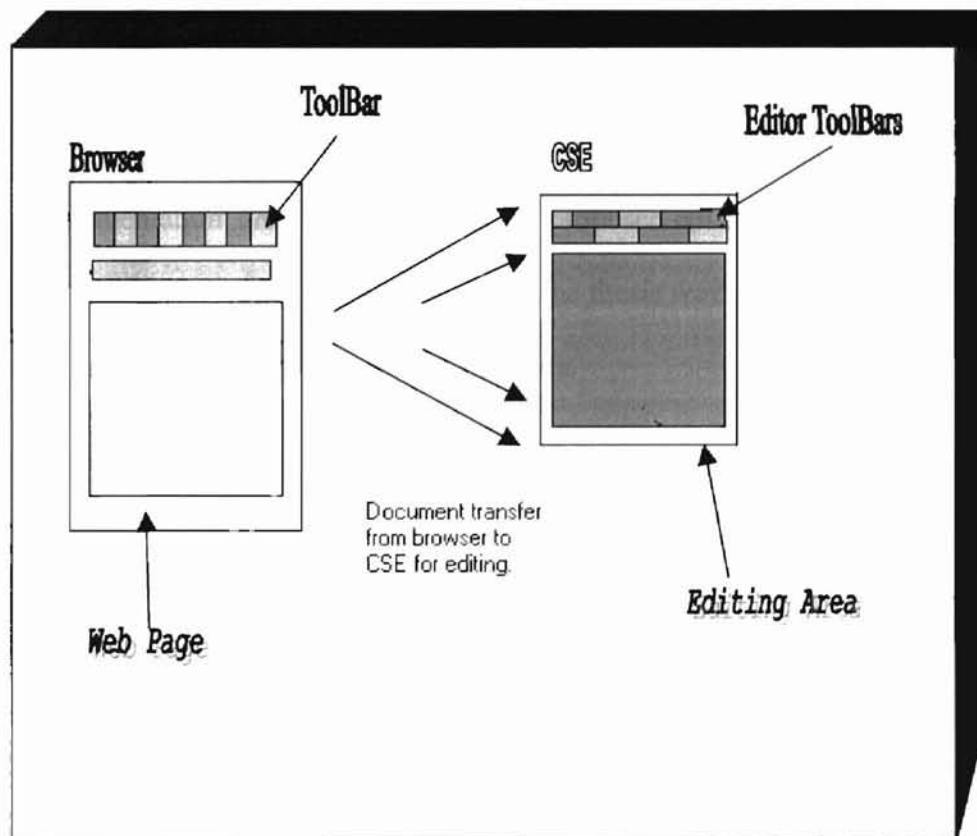


Figure 1.1 CSE Structural Diagram.

1.1 Thesis Organization

The organization of this thesis follows a format that gradually introduces the reader to the work presented as well as related works. The following description outlines each chapter.

Chapter 1, Introduction. This chapter discusses the need for WEB editing tools, and it points out the differences between them. It introduces the need for the editor that we propose, and it illustrates our editor's overall position in the World Wide Web arena.

Chapter 2, Related Works. This chapter covers some of the more popular WEB editing tools available in small detail. It discusses features that draw users to them, and it highlights some of the features that they are missing. The Client-Side Page Editor, CSE, is also introduced in this section.

Chapter 3, CSE Structure & Implementation. This chapter details the design, implementation, and functionality of the proposed editor.

Chapter 4, Conclusion. This chapter concludes the thesis work, and it introduces future possibilities for research in this area.

CHAPTER II

2.0 Related Work

The classification of the Client-Side Page Editor, CSE, as a WEB editing tool opens two areas of discussion concerning related work. The first area includes current WEB editors, and the second includes language constructions currently available to enable users to utilize the CSE's specific features. These subtopics are related directly, but each demands a separate discussion to understand the relation between the two. This chapter includes sections that discuss the structure of current WEB documents, language constructions that implement page breaks in WEB documents, and WEB editors that are available for WEB editing.

2.1 Current WEB Document Structure

Currently, WEB documents are presented to browsers in a format comparable to a "page" in general terms. By comparable, we mean, this format may appear as a page in the viewing area, but, actually, the document is significantly larger than a printed page. By a printed page, we mean a block of data a standard printer can print on a page. A WEB page is defined to be a block of data available on the World Wide Web identified by a URL, Universal Resource Locator. In the simplest, most common case, a web page is a file written in HTML and stored on a server. It may refer to images, which appear as part of the page when a WEB browser displays the document [14]. The problem now becomes defining a feature

that provides users with a more familiar concept of a page with documents on the WEB, or WEB pages. This involves interpreting what constitutes a page and how can we relate this to WEB pages. The simplest idea is to say that a page as defined above, is comparable to one viewing screen via the Internet. This conclusion now leads us in a proper direction. For, we now can define a definition for “page break”. A page break can be defined as a feature that segments portions of a predefined WEB page so that it is comparable to a page in normal terminology. This segmenting should be noticeable in viewing and printing WEB pages.

2.2 Current Page Break Solutions

Due to *limited* language constructs that allow a WEB developer to represent one WEB page from another internally, they must rely on other ways to represent multiple pages in a WEB document [2,4,7,9,10, 18,19,20]. In an attempt to overcome this non-trivial problem, designers have shown an admirable sense of creative ingenuity. Creating anchor points, which are links to specified portions of lengthy documents, appears to be one of the most widely used techniques by WEB developers [8]. The implementation of this concept requires the developer to have a pre-defined layout for the beginning of document segments. The developer must define anchor points to move from one location to the next. They must implement these anchor points as hyperlinks, a cross-reference in an electronic document that, when activated, loads a different section of the document, (figures 2.21 and 2.22) [14]. This idea is a reasonable and

straightforward one; it solves the problem of scrolling through lengthy documents. However, it does not solve one of the problems on which we are focusing; namely, printing. Since these documents continue to be represented in a single file format, the user remains unable to identify portions of a document to be printed once it has been developed [2,4,7,9,10, 18, 19, 20]. To receive specific contents of a WEB document, the user must print the entire document then sift through the output to find the desired material. This can be a waste of printer resources and time.

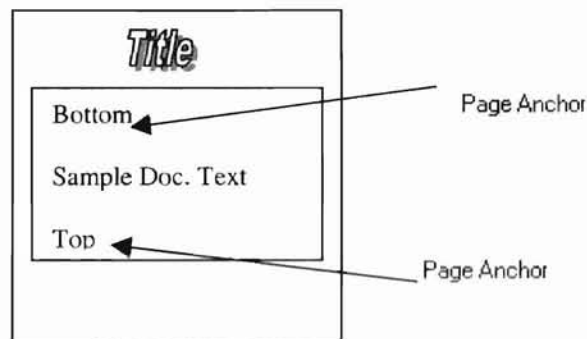


Figure 2.21. Web document displaying anchor technique.

```
<html>
<head> <title> Sample Document </title>
</head>
<body>
<h1 align="center"> Title </h1>
<a name="top">
<a href="#bottom"> Bottom </a>
<br>
<br>
<br>
```

```
Sample Document.....  
<br>  
<br>  
<a name="bottom">  
<a href="#top"> Top </a>  
</body> </html>
```

Figure 2.22. Sample code for figure 2.21.

Cascading Style Sheets, also known as CSS, provide an alternative method to represent segments of a lengthy WEB document, but it is a rarely used feature due to limited exposure. CSS actually defines a “page break” tag that specifies to the printer the beginning and end of a “non-user” defined page. This feature must be inserted directly into the HTML code, and like the previous technique, it implies that the user has no control over where a page begins and ends. The page developer decides the document segmenting points at the time of page creation. By inserting these tags into a document, the printer is able to buffer a document until it reaches the tag location. The printer then prints the contents received as a separate page, and it repeats this process until it reaches the end of the document, see figures 2.23 and 2.24. This is a useful feature, but it has some drawbacks worth mentioning. *It does not allow the user the convenience of specifying desired page break points.* It is recognizable only by printers, not by browsers. It requires more preparation during development, and it must be added as a tag attribute because it is a CSS feature [9]. The latter implies that this break feature will cascade throughout the document; therefore, the designer must pay careful attention to the page’s implementation during development. With the page break provided by the CSS, the user has avoided the problem associated with printing,

but not the one associated with document viewing. Of course, it is possible to implement the two together and alleviate both problems. This requires more preparation and coding, but it is a solution. Nevertheless, the “page break” tag supplied by Cascading Style Sheets and the anchor implementation described are somewhat unattractive because most documents viewed today were created before these features were available. To add either of these to a document would require an unspecified amount of work that not every user would be able to perform. Furthermore, the solutions mentioned must be implemented by the page designer, and the user has no input on the outcome. However, the solution proposed in this thesis provides the user with the opportunity to select the page breaks.

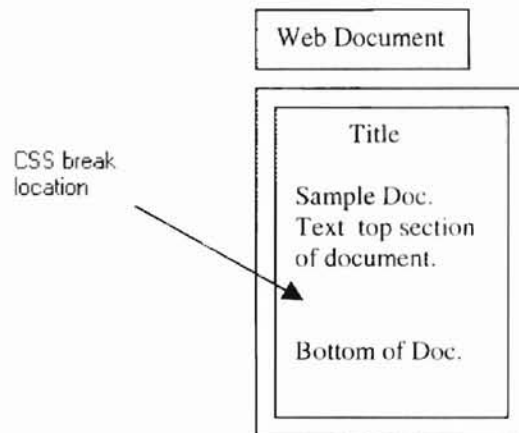


Figure 2.23. WEB view of the CSS page break feature.

```
<html>  
<head>
```

```

<title> Sample Page Testing CSS PAGE BREAK </title>
  <style type="text/css">
    hr.pageEnd { page-break-before: always}
  </style>
</head>
<body>
<h1 align="center"> Title </h1>
<br>
<br>
Sample Document Text. <br> Top Portion of Document <br> <br>
<hr class="pageEnd">
.....
.....
.....
Bottom Section of Sample Document <br> <br>
.....
</body>
</html>

```

Figure 2.24. Sample code for the CSS page break.

2.3 WEB Editors

As mentioned earlier, several editors that allow document editing are available. Some of the more popular window editors include: Microsoft's FrontPage, Sausage Software's HotDog, Netscape's Composer, and many others. All of these editors have features that either make them unique from each of the others or make them more appealing to their users [22].

The first editor mentioned, Microsoft's FrontPage, is a software package for creating and managing WEB sites (figure 2.31 and 2.32). This makes it different from most other editors. Most WEB editors were designed for WEB page creation and editing only. Microsoft's FrontPage provides the user with both of these features and more. Some of its special features include testing the WEB

site's speed, identifying old pages, and repairing broken links. It allows the user to see a directory structure of all pages contained on a site. By running diagnostic reports, FrontPage presents the user with the opportunity to find out which pages are slow, then assist in correcting the problem. Also, it allows the user to identify any out-dated pages helping to maintain a current and up to date site. In addition to the above features, FrontPage is a "what you see is what you get", WYSIWYG, editor that displays exactly what the user enters into the document. This is suitable for many WEB page authors who know little, or nothing, about designing WEB pages. It also can be used to enter HTML code/tags; this feature is useful for the experienced designers who would rather write the code than simply generate generic pages. To aid in this area, Microsoft added a feature that allows the developer to personalize the WEB page code produced. This feature indents the code based on the settings that user provides, it presents the tags in specified colors to distinguish them from one another, and it provides many other personalized settings for developers [24]. Adding to its HTML editing features, FrontPage provides the user with the ability to create, edit, and debug scripts [21]. This is useful to the developers that produce dynamic pages for viewing. As a bonus to all of the above features, it can work directly with Microsoft Office. This allows users to directly publish word documents on their site, and it opens opportunities to have database interactions [24].

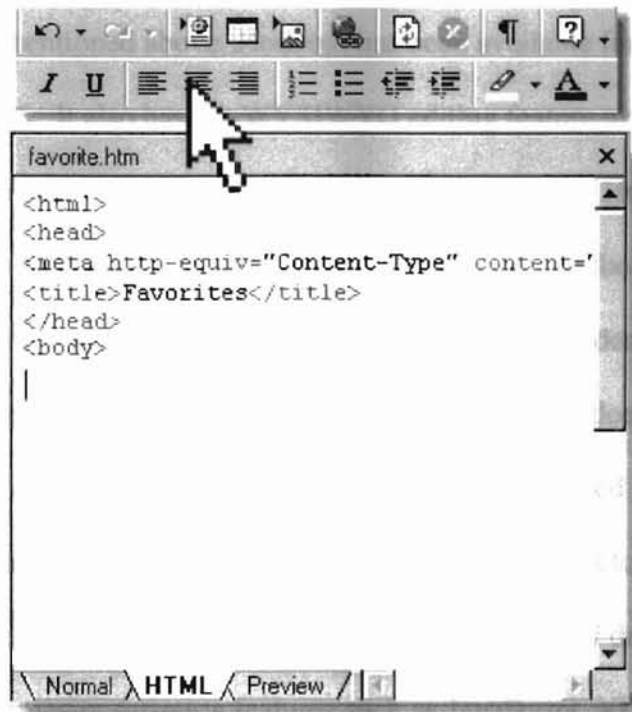


Figure 2.31. Direct HTML coding view.



Figure 2.32. WYSIWYG coding view.

The second editor mentioned above, HotDog, has been available for WEB editing since March 1996. It also has the WYSIWYG editing feature along with the other basic editing features described for FrontPage (figure 2.33). However, it is liked not because of extra features that it has, or does not have, but because of its simple editing environment. The current version of HotDog provides the user with multiple features that allow easy and efficient creation of WEB documents. One of its newest features is its ability to drag and drop icons to desired locations to create documents, almost completely eliminating the need for users to manipulate the HTML code [16]. These point and click, or drag and drop, features now include the newer scripting features such as JavaScript, VB Script, CSS/CSSP, and ASP. These new additions allow the developer to determine the location where the script should be placed, choose which script desired from a collection of script source code, and insert it into the document. This eliminates the need to “re-invent the wheel” while document scripting, and it speeds the development of WEB pages. HotDog’s current editor also provides the user with a “Speedy Document Navigation” feature. This feature allows the user to navigate through a multiple document site in a small amount of time. Through the use of a directory structured tree view, the user can move from file to file with a click of the mouse [17]. In addition to all of its new, user friendly features, HotDog allows the users to use image editing tools, such as Paint Shop Pro, to provide an even better experience during WEB site creation.

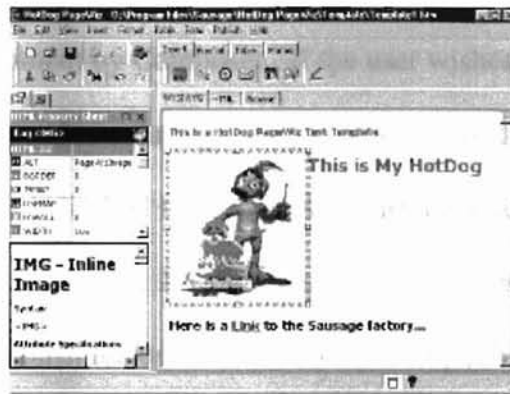


Figure 2.33. HotDog editor view.

The next editor mentioned, Netscape's Composer, is the last of our examples, but it is not the last of the editors available for editing. As are many of the others, it is a very easy to use WYSIWYG editor, once a user becomes familiar with it. But, unlike the others, it is easier to access. Most of these WEB developing tools require the user to download them from various sites, or purchase them, to use. However, Netscape's Composer comes with the Netscape Browser, which was once shipped with the computer system [23]. Today, it can be obtained via a download from Netscape's home site. Its WEB editing environment is similar to that of a word processing application (see figure 2.34). The user enters the document contents as if they were creating a text document, and the published results would be just as they entered it, WYSIWYG. It does not allow the user to enter raw HTML code directly during document development, but it interacts with

other applications; i.e., a HTML source editor, to allow the user to enter raw HTML code (figure 2.35). It supports all of the basic HTML language constructions, and it allows the user to enter extra code statements to give the WEB page some added features. Most of these constructions are created and inserted into the document by Composer. If the user wishes to enter HTML code directly, then they must do so through the use of a specified source editor. Also, Netscape's Composer is limited in its support of newer concepts such as CSS/CSSP, DHTML, and some JavaScript syntax structures. Like Microsoft's FrontPage, it has a publish feature that assists users in publishing their document on a specified server. This feature updates the links contained in the document so that they refer to the relative path of the document's location on the server. It automatically uploads any images and pages associated with each document, onto the server for publishing.

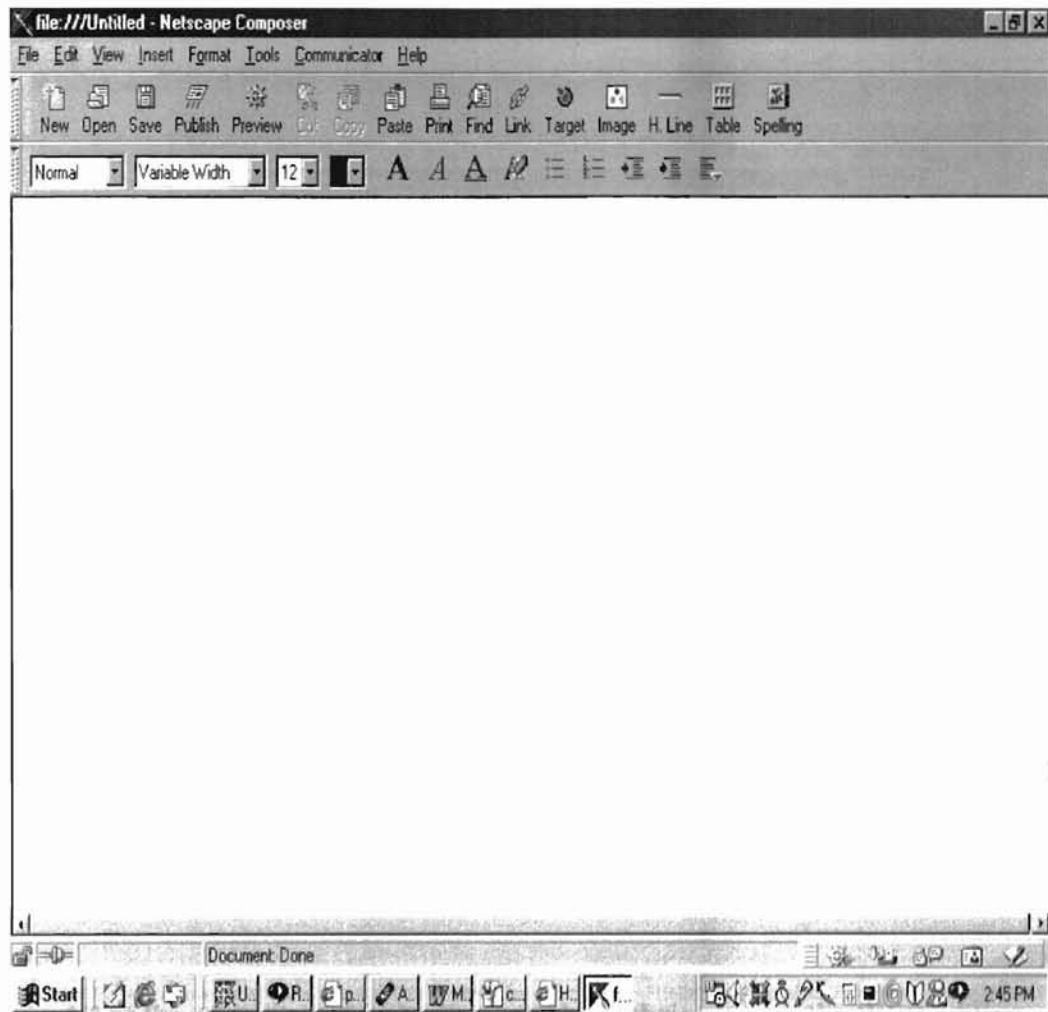


Figure 2.34. Netscape Composer's editing area.

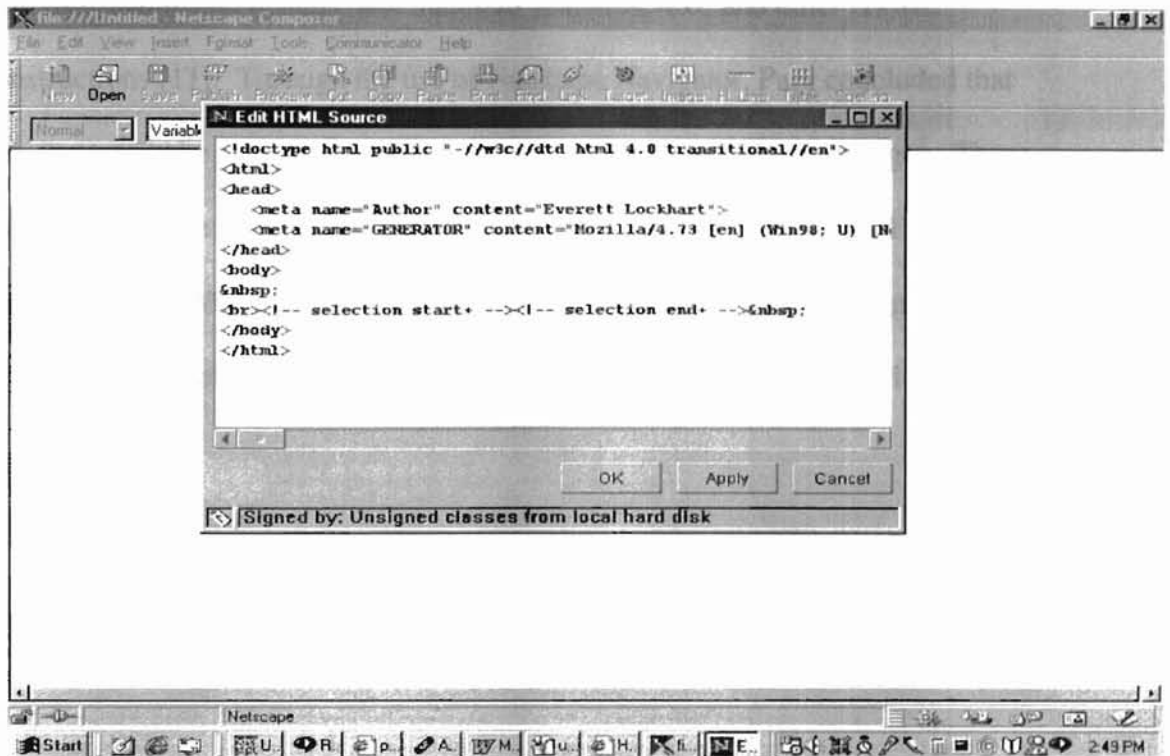


Figure 2.35. Composer with source code editor open.

All of the editors previously discussed have their place in the WEB editing community, and they satisfy multiple needs for the user. As with any software product, they each have their drawbacks. However, the most visible aspect is that none have the pre-mentioned page break feature.

In earlier efforts to solve the problem stated earlier, Joongseok Park introduced a method of segmenting a document into sections to be broken into pages. By sectioning text through the use of the “<DIV>” HTML tags, Park concluded that an editor could be implemented that recognizes these tags as beginning and ending page boundaries (figure 2.36). This tag was inserted into the

HTML code via the document's code view. His work shows that an editor defined with this feature can manage and maintain graphics, text, and other HTML constructions [11]. Through the use of Netscape Navigator, Park concluded that an editor could be implemented using current WEB developing languages. These languages include: Cascading Style Sheets (CSS), Dynamic HTML (DHTML), and JavaScript. With the languages mentioned, the editor could be designed as a client-side application that has dynamic text-modification capability [11].

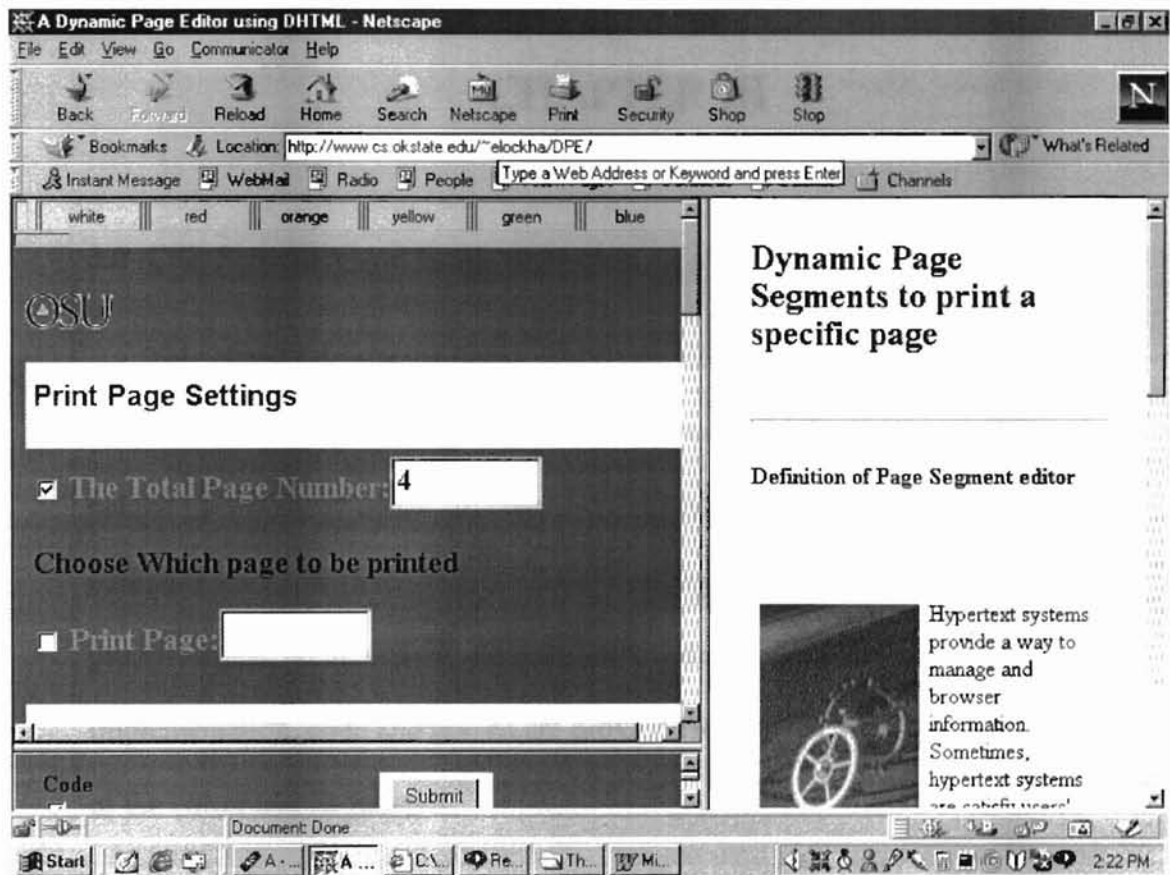


Figure 2.36. Dynamic Page Editor (DPE) editing view.

CHAPTER III

3.0 CSE Structure and Implementation

This chapter details the design overview along with the implementation details of the page editor implemented in this research. It explains the reasoning behind the choice of editor structure, coding style, and languages used. Various editing features are presented, but the overall focus is directed towards the proposed page break feature. In addition to a detailed explanation of the implementation, code and graphs are provided to provide a visual representation of the editor and its features.

3.1 Design

In the early stages of development, it was necessary to determine an editor framework to allow the manipulation of a document's text features and to present the users with a familiar WEB browsing atmosphere. To accomplish these goals, a set of design criteria must be determined. The design criteria for the CSE include the following: a) the editor's base structure should contain, or have the capability of supporting, exterior features to allow document text manipulation, b) the structure should support the pre-defined page break tag used, and c) it should be user friendly. To satisfy the conditions mentioned, Internet Explorer's, IE, base structure was chosen as our infrastructure, *see Figure 3.1*.

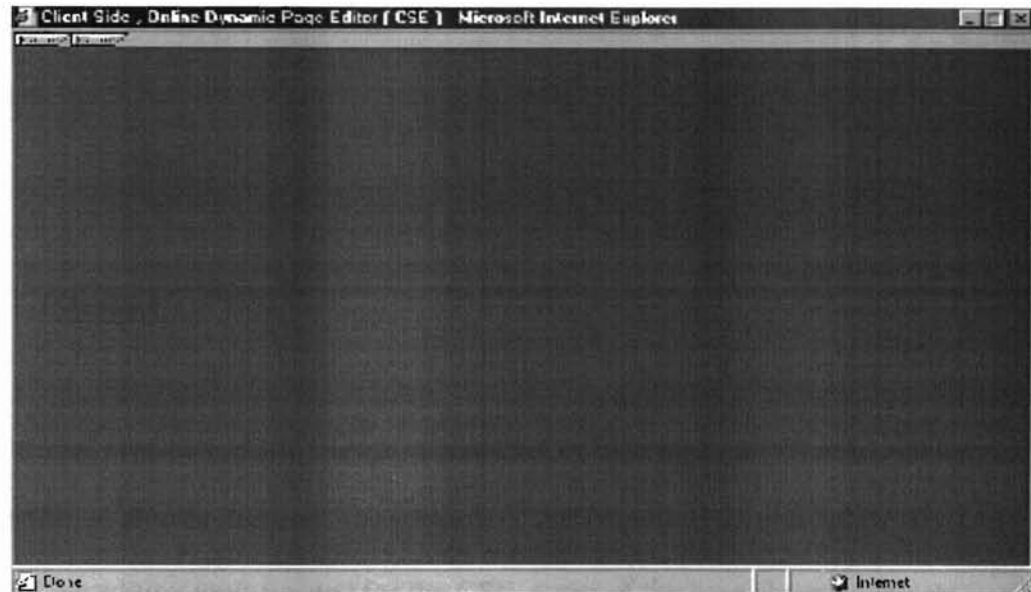


Figure 3.1 Client-Side Page Editor with IE structure.

This structure gives the application the look and feel of a very familiar browser. It enables the utilization of text-editing features that are not readily available in other browsers. In addition to the special text-editing features provided by Microsoft through the use of Internet Explorer 5.5+, it is also possible to utilize specific features of DHTML and CSS/CSSP that are not available in Netscape's Navigator.

Once the application's structure was determined, the next step is to determine the available tools for implementing the functionality of the page break feature. Various constructions are available for WEB editing and text manipulation during document creation; however, it is necessary to determine whether it is possible to manipulate the document's features by someone other than the developer [3]. This manipulation includes basic text editing features as well as the insertion of

the editor defined page break. Again, the overall focal point is directed towards the page break feature with references to the basic editing features defined for the CSE.

3.2 Interface

The base structure of the CSE is an instance of Internet Explorer with limited and modified chrome features, i.e. editor toolbars (*figures 3.21 and 3.22*). To provide the editing tools needed for the CSE, some of the base chrome features are omitted and replaced with editor defined toolbars. The new toolbar features allow the user to utilize the features provided by the CSE.

The implementation of the editor's interface includes a number of stages to reach completion. The first stage includes the creation of the viewing area, the initial window. In creating the window structure, the default chrome features are omitted to provide a more spacious area into which is placed the editor-defined toolbars. It is also necessary to omit some of the base features to control what the user can and cannot do when using the editor.

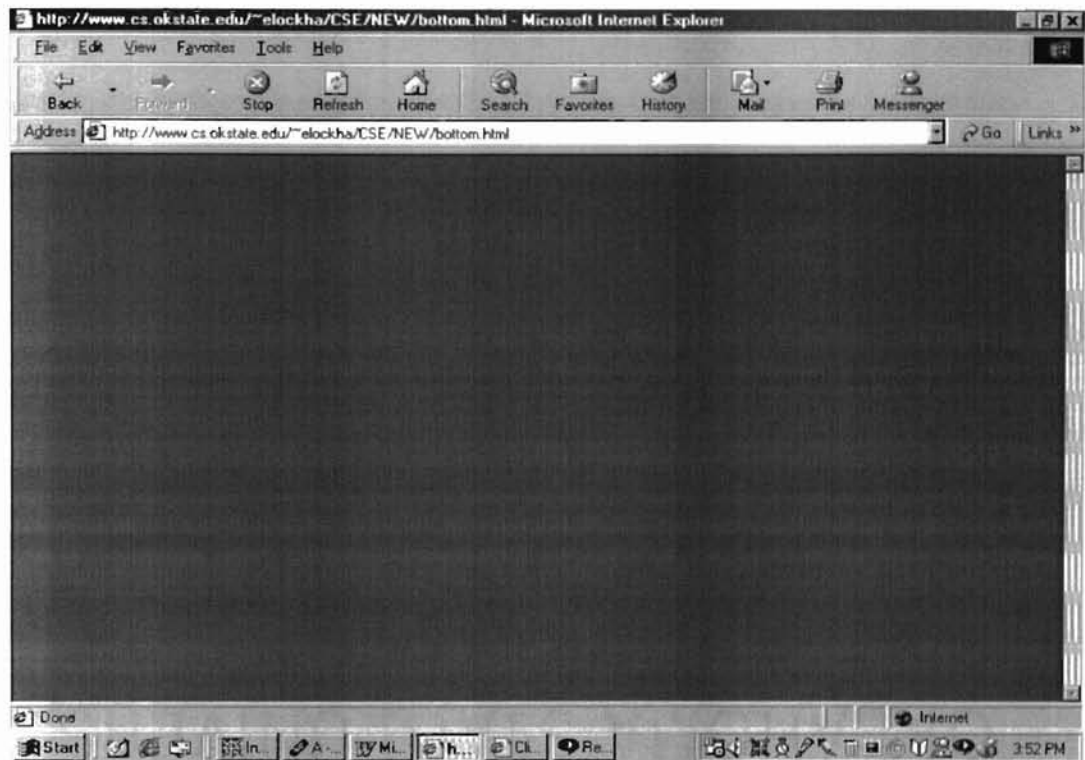


Figure 3.21 Internet Explorer with default chrome features.

The new toolbar features are created dynamically for each client when the editor is loaded into the viewing area. Toolbar positioning is set during document loading using CSSP, and the actual toolbar is created using JavaScript code [15]. Once the toolbar has been created and positioned, the editor components are linked to their corresponding toolbar buttons (*figures 3.24 and 3.25*). This provides the user with the ability to interact with the editor and its features (*figure 3.22*).

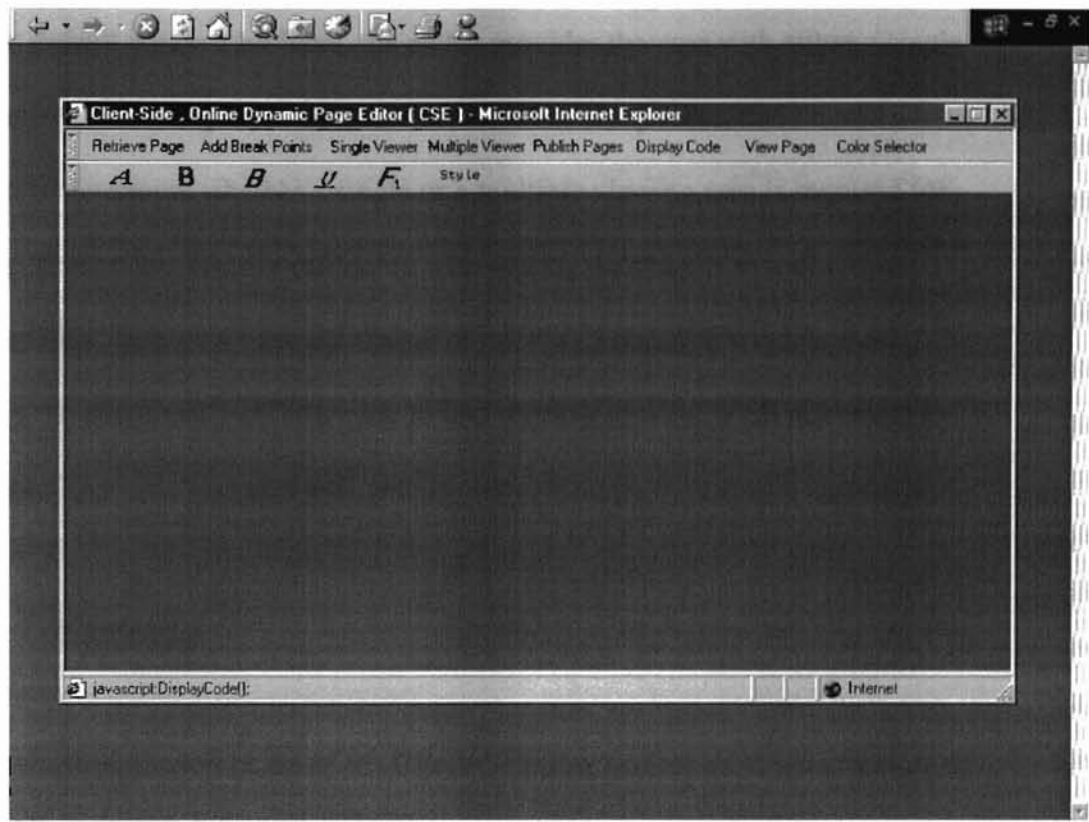


Figure 3.22. CSE interface with editor defined toolbars.

```
<STYLE TYPE="text/css">
    #toolbar0 { position:absolute; }
    #text0 { position:absolute; }
    #toolbar1 { position:absolute; }
    #text1 { position:absolute; }
    #toolbarCol { position:absolute; }
</STYLE>
```

Figure 3.24. Code segment for position toolbars [15].

```
var edit_bar = new Toolbar(document.all.toolbar1, document.all.toolbarCol);
editor_comp.Name = "Editor Tool Bar"
editor_comp.addItem("Retrieve Page", "javascript:GetNewPage();");
editor_comp.addItem("Add Break Points", "javascript:InsertBreaks();", 100);
```

Figure 3.25. Code segment for creating toolbars [15].

In addition to the editor-defined toolbar features that are created to replace specific default window toolbars, the editor provides the user with either a single or a multi-view editing area. Included as a toolbar component, this feature allows the user to determine whether a single or a multiple viewing area is required for editing. If users are more comfortable with editing documents in multi-view environments, then they have the option of opening a separate window that displays the document's HTML code in one window and the WEB document in the other. This option is also available for users who prefer a single view option.

3.3 Environment

The implementation phase of the Client-Side Page editor includes a mixture of JavaScript, DHTML, and CSS programming features, which currently are available only in Internet Explorer versions 4.0 and higher [6]. A few problems deserve special attention to implement the page break feature. *Is it possible to capture the text of a WEB document by a user other than the developer? If so, how could it be done without sending additional requests to the WEB page's domain?* These questions are addressed in the following sections.

3.4 Capturing And Modifying Document Text

In an attempt to dynamically insert a page break into a WEB document, one goal of this project is to perform this task without sending extra requests to the WEB page's domain. This goal is important in reducing the editor's response

time when the user chooses the area where a break is to be inserted. Through the use of DHTML language constructions, the editor can accomplish this task.

Microsoft's Internet Explorer supports Dynamic HTML (DHTML) functions that allow the capturing and modification of a document's text by clients. The HTML object of the DHTML language has a predefined method, `createTextRange()`, that provides the editor with part of the functionality stated above. This feature allows the CSE physically to obtain the document's text after it is loaded into the client's viewing area. The task becomes a function of recognizing and manipulating the area specified as a break location. The regular expression object and `ondblclick()` event handler, which are features of the JavaScript language, provide the editor with these capabilities. In addition to the insertion of page breaks, these features provide the CSE with the basic editing features visible in most text editors. These features include the modification of the text's style, font size, and font type.

3.5 Functionality

The external and internal operations of the CSE require an overview so that the user is able to operate it properly. After the user opens the editor for usage, the URL of the page to be edited should be entered and retrieved. Once the document is loaded, the user is free to decide break point locations. To enter a break point, the user double clicks on a highlighted area desired as a break point. Once this is done, the document's text is captured, then a regular expression is

compiled to recognize the user-specified break location (Figure 3.50). After the location is recognized, the editor inserts the editor-defined breaks into the document's new page representation. The new page is generated dynamically by the CSE on the client machine; it contains the break points that the user specifies to be part of its contents. Once the user has selected and entered all possible break-point locations, the original document is broken into separate pages by clicking on the "publish button" located on the editor's toolbar. After all steps have been completed, the document is reproduced as multiple pages based on the user's break point choices. All of this can be done while the user continues to browse the Internet.

```
reg_exp.compile ( "\\B<\\s*hr\\s+...\\s+color=black\\s+id="+Event+" ...>\\B");
```

Figure 3.50. Regular Expression to recognize break location.

3.6 Security

Due to the increasing concern about Internet security, it has become necessary to give this concept careful consideration when implementing applications for use over the Internet. The CSE is used both to edit and to modify WEB pages for viewing; therefore, it could be classified as a *helper application*. A "helper application" is an application that allows the user to manage files that are stored as data types other than the standard types recognized by most browsers. These

applications bring attention to WEB security when used via the Internet. Some of the most well-known helper applications include: word processing applications, spread sheets, and any other applications that aid the user in viewing information stored as non-standard WEB data types [5]. Because most of these applications are written in languages such as Visual Basic, Visual C++, etc., they place the user's computer at risk of attacks from malicious programs. Through the use of these applications, an attacker can persuade the user to download a program that can use these helper applications to execute programs that will attack their computer in various ways [5]. The CSE is written in languages that reduce some of the security risks opened by other applications. The chosen languages used do not allow either the reading or writing of files to the user's system; therefore, it reduces some of the major hacking possibilities. Through the use of the CSE, clients have a user-friendly, page editor that does not open their computer to additional security violations.

Chapter IV

4.0 Conclusion & Future Work

The CSE provides Internet users with a basic feature needed to edit WEB documents, a page break. In its current state, it does not provide the user with the full functionality of an editor. However, it is possible to include all features necessary to complete this task. The CSE provides the user with all of the comforts of other editors currently available. It gives the added option of breaking the document into multiple pages as well. It permits the user to have complete control over where a break occurs, and it allows this to be done on the client's side of the Internet. Expanding the editor's base functionality could include the addition of multiple features. One feature could be the generic inclusion of breaks throughout a document during the initial load. Others could include an option to load the current document being viewed, or to retrieve a new document.

Bibliography

- [1] Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. Compilers –Principles, Techniques, and Tools. Addison-Wesley Publishing. United States of America. 1986.
- [2] Bradenbaugh, Jerry. JavaScript Application Cookbook. O'Reilly & Associates, Inc. Sebastopol, CA. 1999.
- [3] Bull, G., Gina Bull, and Dave Lewis. "Introducing Dynamic HTML." *Learning and Leading with Technology*. 26.2 (1998): 43-45. CML 2000 School. EJ577901. 31 October 2000.
- [4] Flanagan, David. JavaScript "The Definitive Guide", 3rd edition. O'Reilly & Associates, Inc. Sebastopol, CA. 1998,1997, 1996.
- [5] Garfinkel, Simson, and Gene Spafford. Web Security & Commerce. O'Reilly & Associates, Inc. Sebastopol, CA. 1997.
- [6] Goodman, Danny. Dynamic HTML - The Definitive Reference. O'Reilly & Associates, Inc. Sebastopol, CA. 1998.
- [7] Goodman, Danny. JavaScript Bible, 3rd edition. IDG Books Worldwide, Inc. Foster City, CA. 1998.
- [8] Kerven, David, Jeff Foust, and John Zakour. HTML 3.2 plus – The Definitive HTML 3.2 Problem Solver. Mitchell Waite. Corte Madera, CA. 1997.
- [9] Livingston, Dan, and Micah Brown. Essential CSS & DHTML For Web Professionals. Prentice-Hall, Inc. Upper Saddle River, NJ. 1999.
- [10] Moncur, Micheal, and Laura Lemay. JavaScript. Sams.net Publishing. Indianapolis, IN. 1996.

- [11] Park, Joongseok. *"A Dynamic Page Editor."* M.S. Thesis. Oklahoma State University. 1999.
- [12] Spainbour, Stephen, & Robert Eckstein. Webmaster In A Nutshell - A Desktop Quick Reference, 2nd edition. O'Reilly & Associates, Inc. Sebastopol, CA. 1999.
- [13] ClientSide JavaScript Reference.
<http://netscape.com/docs/manuals/js/client/jsref/index.htm>.
- [14] Dictionary
<http://www.dictionary.com/>.
- [15] DHTML Toolbars: An Interface Users Will Recognize.
<http://developer.netscape.com/docs/technote/dynhtml/toolbar/index.html>.
- [16] HotDog Express
<http://detective1.com/hotdogXp.htm>.
- [17] HotDog Professional
<http://www.sausage.com/>.
- [18] JavaScript Application Cookbook.
<http://www.server.com/hotstyle/>.
- [19] JavaScript Documentation.
<http://netscape.com/docs/manuals/javascript>.
- [20] JavaScript "The Definitive Guide", 3rd edition.
<http://www.oreilly.com/modbin/books.mod/webref/updates/jscript3/index.htm>.
- [21] Microsoft FrontPage 2000 Evaluation
<http://www.microsoft.com/frontpage/evaluation.htm>.

[22] There is A Perfect Editor.

<http://wysiwyg://153/http://www.csn.net/~bediger/perfect.editor.html>.

[23] Using Netscape Page Composer.

<http://online.parkland.cc.il.us/presentations/netscapepc/>

[24] Microsoft FrontPage Tour

<http://www.microsoft.com/frontpage/2000/fp2kPg1.htm>

VITA

Everett Lockhart

Candidate for the Degree of

Master of Science

Thesis: CLIENT SIDE PAGE EDITOR

Major Field: Computer Science

Biographical:

Personal Data: Born in Montgomery, Mississippi, On September 24, 1974, the son of Henry and Sandra Brantley.

Education: Graduated from Kosciusko High School, Kosciusko, Mississippi in May 1993; received Bachelor of Science degree in Mathematics with an emphasis in Computer Science from Tougaloo College, Tougaloo, Mississippi in May 1997. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 2000.

Experience: Course lecturer for C and Unix programming classes, Head-Teaching Assistant for Visual Basic programming course, Teaching assistant for Java and Pascal programming courses, lecturer for summer science web development program, and researched parallel data structures two consecutive summers at Oklahoma State University.

Professional Memberships: Alpha Chi and Alpha Lambda Delta honor societies, Kappa Alpha Psi Fraternity, Inc.